18-default vs parameterized constructor

In Java, constructors are special methods used to initialize objects. When an object of a class is created, the constructor is automatically called. Constructors can either set default values or accept parameters to set custom values for object attributes.

Types of Constructors in Java

Constructors in Java can be broadly categorized into two types:

1. **Default Constructor:**
    - A constructor that has no parameters.
    - It is automatically provided by the Java compiler if no constructor is explicitly defined.
    - **Syntax:**

<class_name>() {}

    - If a class has no constructors, the compiler automatically creates a default constructor.
2. **Parameterized Constructor:**
    - A constructor that accepts parameters to initialize an object with specific values.
    - Allows for greater flexibility in object creation.
    - **Example:**

```
public Human(int age, String name) {
    this.age = age;
    this.name = name;
}
```

Key Differences Between Default and Parameterized Constructors

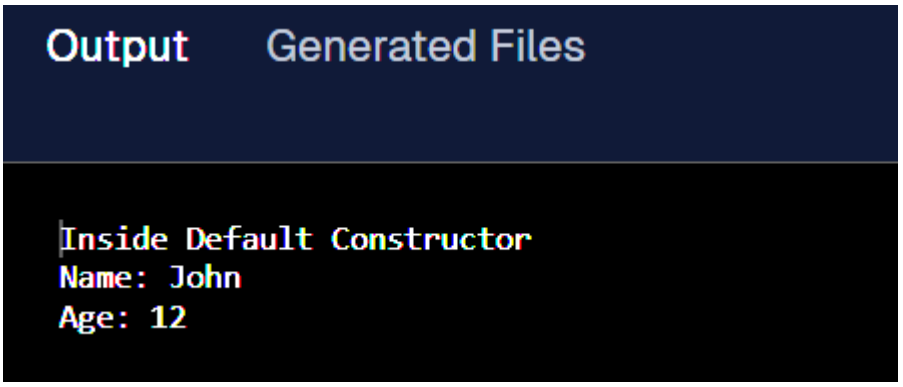
- **Default Constructor:**
    - Has no parameters.
    - Provides default values to object attributes.
    - Automatically generated by the compiler if not explicitly defined.
- **Parameterized Constructor:**
    - Accepts arguments to initialize object attributes with specific values.

- Must be explicitly defined by the programmer.

Example: Default Constructor

```java
class Human {
    private int age;
    private String name;

    public Human() {  // Default Constructor
        System.out.println("Inside Default Constructor");
        age = 12;
        name = "John";
    }

    public int getAge() {
        return age;
    }

    public String getName() {
        return name;
    }
}

public class Demo {
    public static void main(String[] args) {
        Human obj = new Human();  // Default Constructor is called
        System.out.println("Name: " + obj.getName() + "\nAge: " + obj.getAge());
    }
}
```

Output:



In this example, every time an object of the Human class is created using the default constructor, the same default values (age = 12, name = "John") are assigned to the object's attributes.

Example: Parameterized Constructor

```java
class Human {
    private int age;
    private String name;

    public Human() {  // Default Constructor
        System.out.println("Inside Default Constructor");
        age = 12;
        name = "John";
    }

    public Human(int a, String n) {  // Parameterized Constructor
        System.out.println("Inside Parameterized Constructor");
        age = a;
        name = n;
    }

    public int getAge() {
        return age;
    }

    public String getName() {
        return name;
    }
}

public class Demo {
    public static void main(String[] args) {
        Human obj = new Human();  // Default Constructor is called
        Human obj1 = new Human(18, "June");  // Parameterized Constructor is called
        System.out.println("Name: " + obj.getName() + "\nAge: " + obj.getAge());
        System.out.println("Name: " + obj1.getName() + "\nAge: " + obj1.getAge());
    }
}
```

Output:

```
Output    Generated Files

Inside Default Constructor
Inside Parameterized Constructor
Name: John
Age: 12
Name: June
Age: 18
```

Here, the default constructor assigns values for obj, while the parameterized constructor allows different values to be assigned to obj1.

Additional FAQs on Java Constructors

1. **What happens if I define a constructor with parameters and no default constructor?**
    - If you define a parameterized constructor and omit the default constructor, the compiler will not provide a default constructor. Therefore, attempting to create an object without arguments will result in a compile-time error.
2. **Can a constructor be private?**
    - Yes, a constructor can be private. This is often used in the Singleton design pattern to restrict object creation and ensure only one instance of a class exists.
3. **Can constructors be overloaded in Java?**
    - Yes, constructors can be overloaded by defining multiple constructors with different parameter lists within the same class.
4. **Why do we need a parameterized constructor?**
    - A parameterized constructor is useful when you want to create objects with different initial values, providing more control over object creation.